Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютеров и операционные системы

Чистов Даниил Максимович

Содержание

# 1 Цель работы

Цель данной работы - приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

# 2 Задание

* Реализация циклов в NASM
* Обработка аргументов командной строки
* Задание для самостоятельной работы

# 3 Выполнение лабораторной работы

## 3.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8, затем создаю файл lab8-1.asm (рис. [1](#fig:001)).

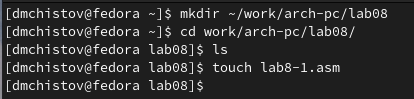


Figure 1: Подготовка к выполнению заданий

Открываю созданный мной файл в mcedit и вставляю код из листинга 8.1 (рис. [2](#fig:002)).

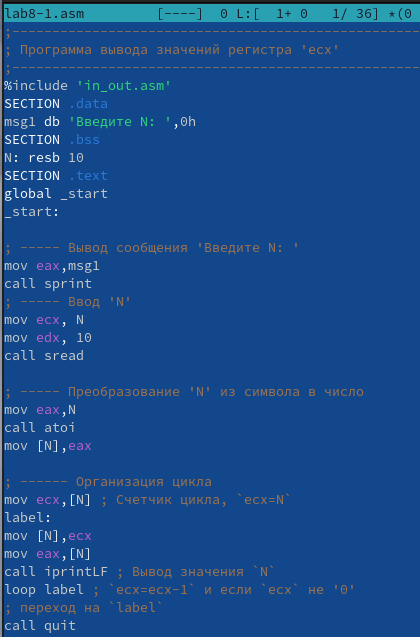


Figure 2: Код программы листинга 8.1

Создаю исполняемый файл и проверяю его работу (рис. [3](#fig:003)).

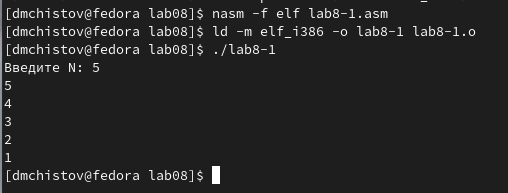


Figure 3: Успешно рабочая программа

Изменяю код, воспользовавшись кодом, приведённом в задании (рис. [4](#fig:004)), затем проверяю его работу (рис. [5](#fig:005)).

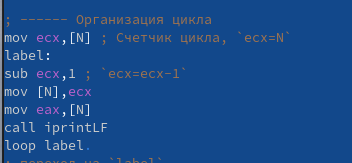


Figure 4: Новый код программы

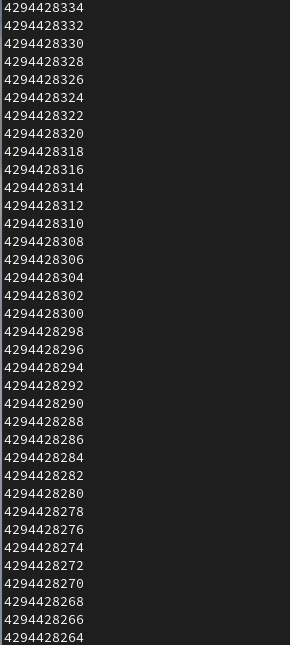


Figure 5: Программа работает некорректно

Данный пример показал, что использование регистра ecx в теле цикла loop может привести к некорректной работе программы. Программа просто выводит бесконечный набор чисел.

Теперь попытаюсь корректно испольщовать регистр ecx, воспользовавшись предоставленным заданием кодом. (рис. [6](#fig:006)) и также проверяю корректность работы программы (рис. [7](#fig:007)).

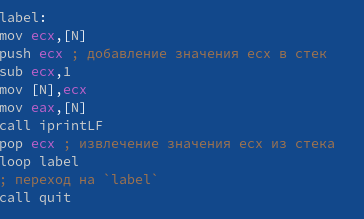


Figure 6: Изменённый код программы

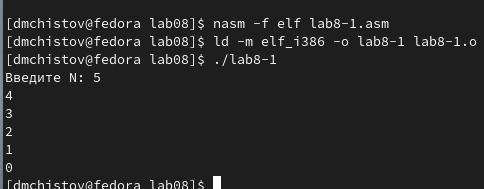


Figure 7: Программа работает корректно

Успешно! В данном случае количество выводой соотвествует с введёным числом N.

## 3.2 Обработка аргументов командной строки

Внимательно изучаю код из листинга 8.2, после чего создаю файл lab8-2.asm (рис. [8](#fig:008)), вставляю код (рис. [9](#fig:009)), создаю файл и указываю следующие аргументы: аргумент1 аргумент 2 ‘аргумент 3’ (рис. [10](#fig:010))

Figure 8: Создание файла

Figure 8: Создание файла

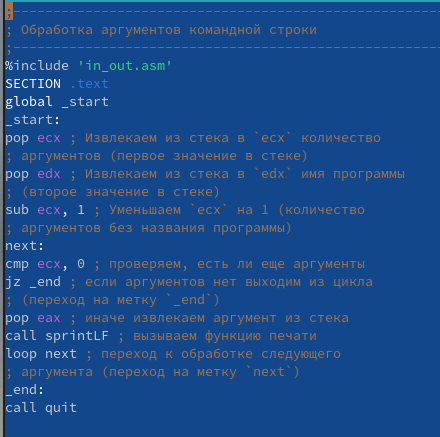


Figure 9: Код программы lab8-2

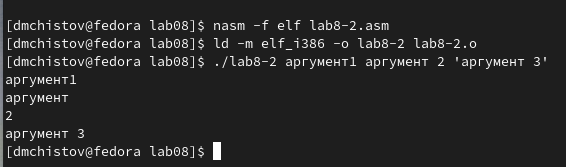


Figure 10: Успешная работа программы

Программа работает успешно!

Теперь создаю новый файл - lab8-3.asm (рис. [11](#fig:011)), и ввожу в него код из листинга 8.3 (рис. [12](#fig:012)).

Figure 11: Создание файла

Figure 11: Создание файла

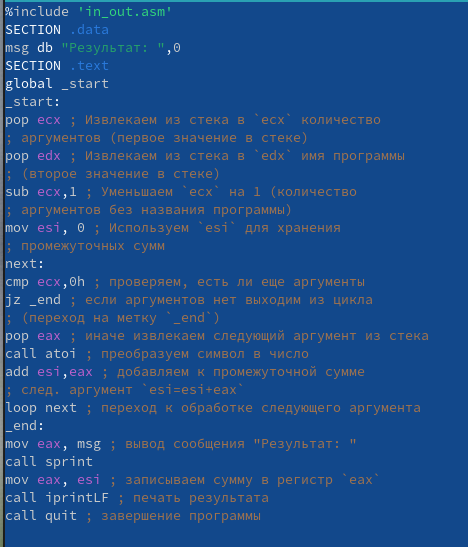


Figure 12: Код программы lab8-3

Создаю файл и указываю аргументы такие же, как в приведённом примере: 12 13 7 10 5 (рис. [13](#fig:013)).

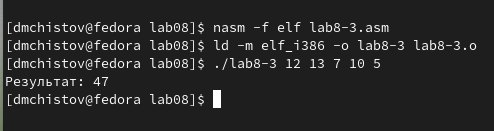


Figure 13: Проверка работы программы

Программа работает успешно!

Теперь от меня требуется изменить код программы 8-3 таким образом, чтобы вместо суммирования аргументов было выполнено произведение. Для этого воспользуюсь обозначением mul, а также перед этим изменю значение esi с 0 на 1, таким образом программа будет умножать каждый аргумент на следующий. Приведу изменённый код программы (рис. [14](#fig:014)).

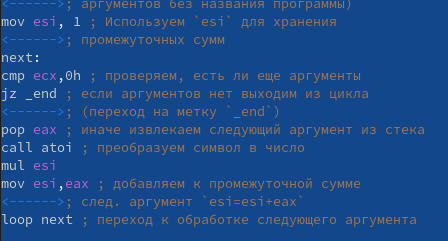


Figure 14: Новый код программы для произведения

Проверяю корректность работы программы (рис. [15](#fig:015)).

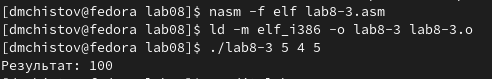


Figure 15: Программа работает успешно

## 3.3 Задание для самостоятельной работы

От меня требуется написать программу, которая суммирует значения определённой функции f(x). Так как мой вариант - 2, моя функция выглядит следующим образом:

Вот готовый код программы в виде скриншота (рис. [16](#fig:016)).

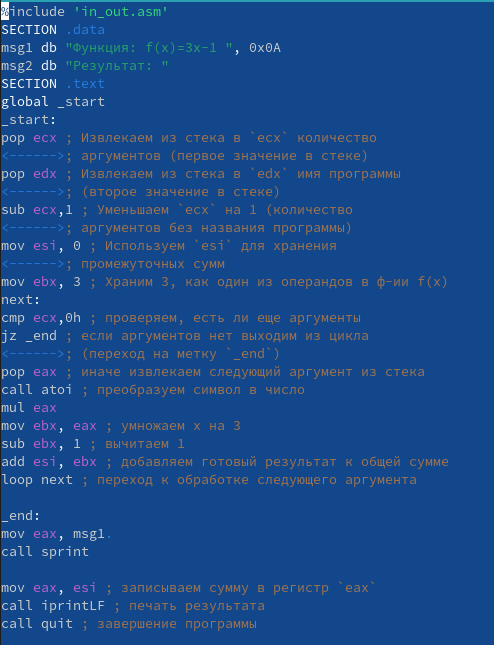


Figure 16: Код программы для задания

Вот код в письменном виде:

%include ‘in\_out.asm’

SECTION .data

msg1 db “Функция: f(x)=3x-1”, 0x0A

msg2 db “Результат:”

SECTION .text

global \_start

\_start:

pop ecx ; Извлекаем из стека в ecx количество

; аргументов (первое значение в стеке)

pop edx ; Извлекаем из стека в edx имя программы

; (второе значение в стеке)

sub ecx,1 ; Уменьшаем ecx на 1 (количество

; аргументов без названия программы)

mov esi, 0 ; Используем esi для хранения

; промежуточных сумм

mov ebx, 3 ; Храним 3, как один из операндов в ф-ии f(x)

next:

cmp ecx,0h ; проверяем, есть ли еще аргументы

jz \_end ; если аргументов нет выходим из цикла

; (переход на метку `\_end`)

pop eax ; иначе извлекаем следующий аргумент из стека

call atoi ; преобразуем символ в число

mul eax

mov ebx, eax ; умножаем x на 3

sub ebx, 1 ; вычитаем 1

add esi, ebx ; добавляем готовый результат к общей сумме

loop next ; переход к обработке следующего аргумента

\_end:

mov eax, msg1

call sprint

mov eax, esi ; записываем сумму в регистр eax

call iprintLF ; печать результата

call quit ; завершение программы

Теперь нужно проверить корректность работы моего кода (рис. [17](#fig:017)).

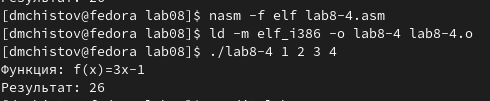


Figure 17: Программа работает успешно!

# 4 Выводы

Я выполнил поставленные цели лабораторной работы, благодаря этому я приобрёл навыки написания программ с использованием циклов и обработкой аргументов командной строки.

# Список литературы

[Лабораторная работы №8](https://esystem.rudn.ru/pluginfile.php/2089095/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%968.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D1%86%D0%B8%D0%BA%D0%BB%D0%B0.%20%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%B0%D1%80%D0%B3%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%B2%20%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%BE%D0%B9%20%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8..pdf)